EE735 Computer Vision
Programming Assignment #3

Semi-supervised Image Classification with Deep Neural Networks

TA. Dong-Jin Kim

[1] Semi-Supervised Learning with Ladder Networks. NIPS 2015.
[2] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.
[3] MEAN TEACHERS ARE BETTER ROLE MODELS: Weight-averaged consistency targets improve semi-supervised deep learning results. NIPS 2017.
[4] Smooth Neighbors on Teacher Graphs for Semi-supervised Learning. CVPR 2018.

Supervised Learning VS Unsupervised Learning

Supervised Learning : when all the data points are labeled

Unsupervised Learning : non of the data points are labeled

Supervised Learning VS Unsupervised Learning

Supervised Learning : when all the data points are labeled

Unsupervised Learning : non of the data points are labeled

Semi-supervised = supervised + unsupervised

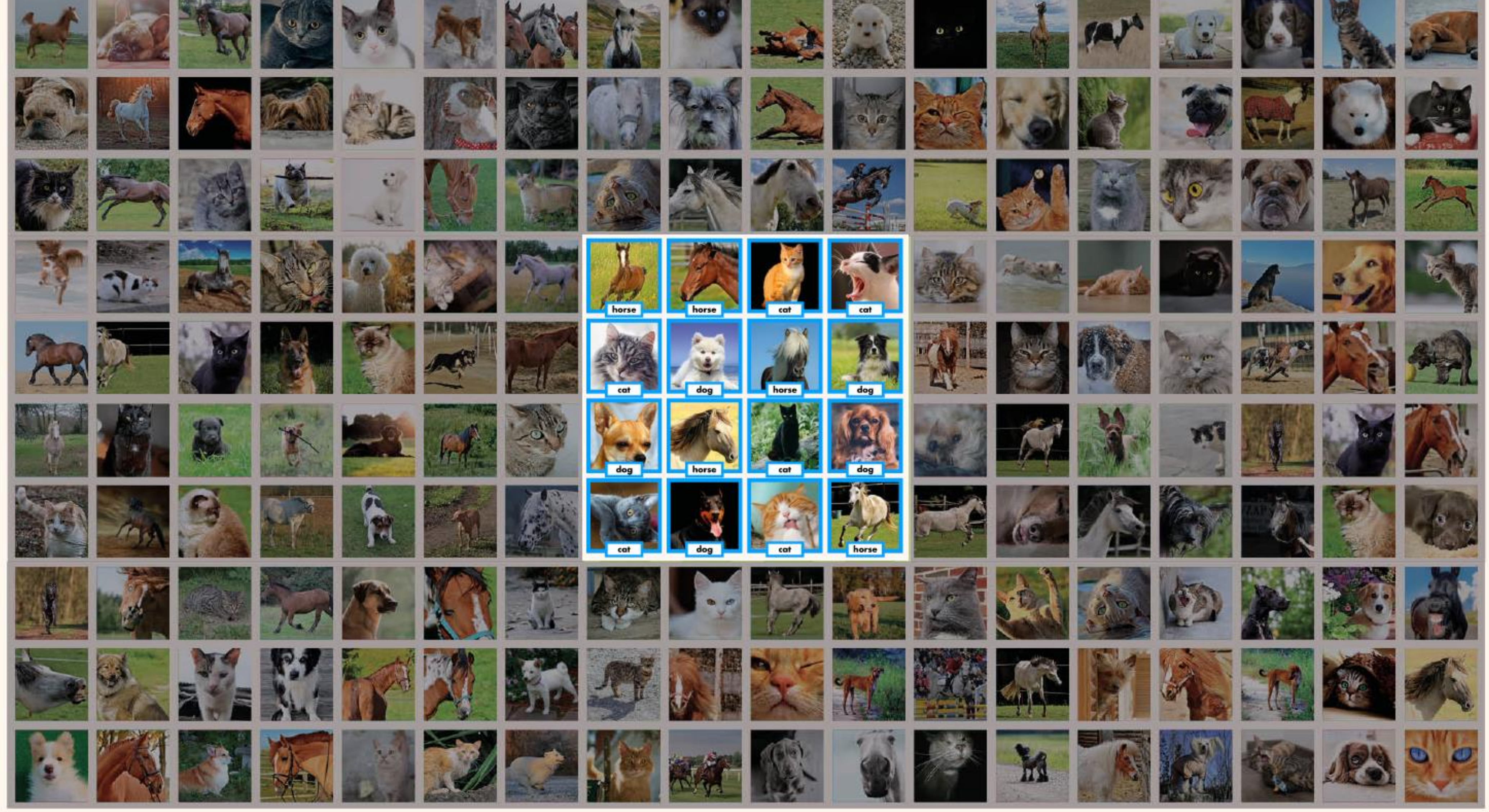


Illustration in the Probability Space

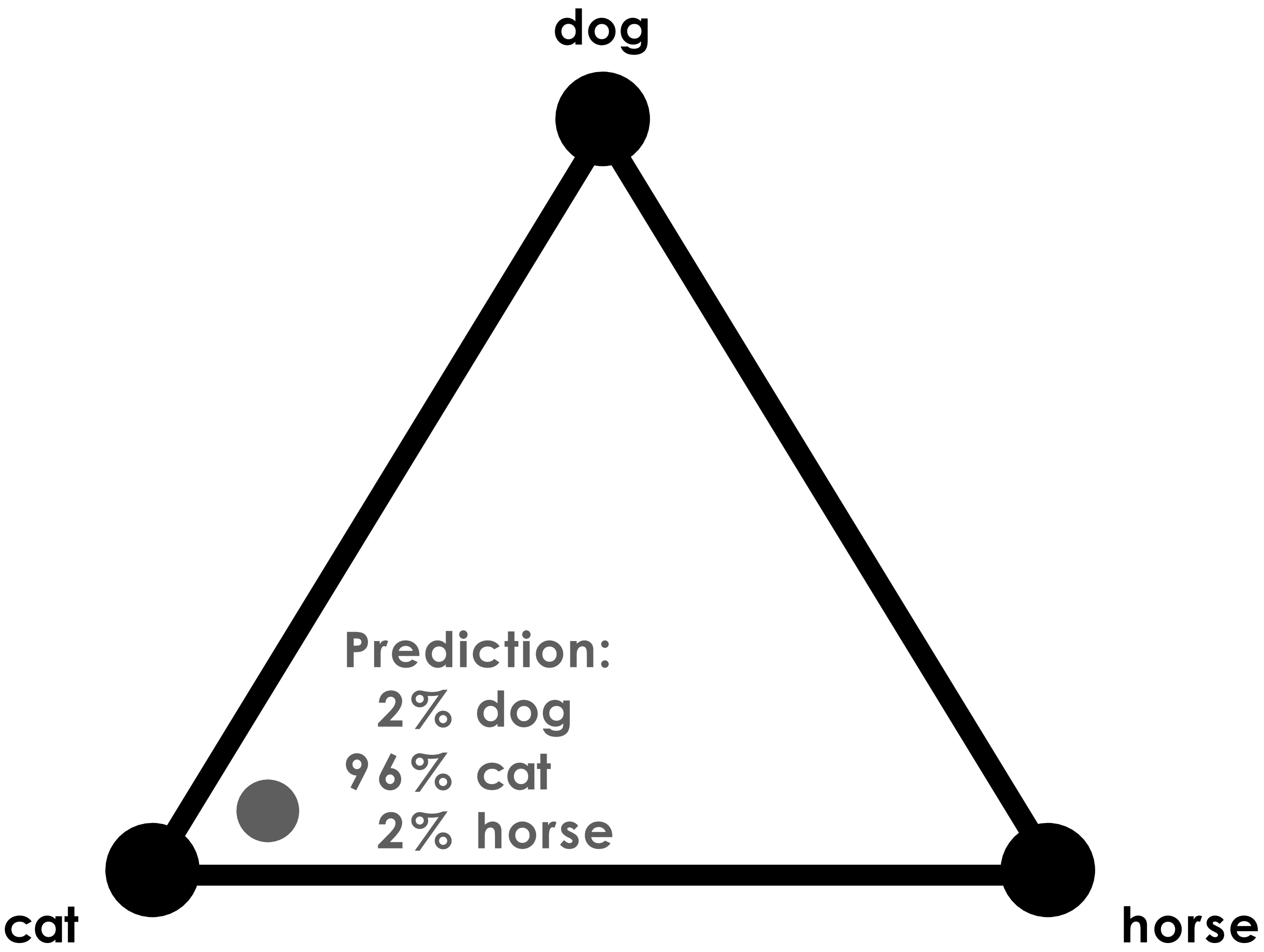


Illustration in the Probability Space

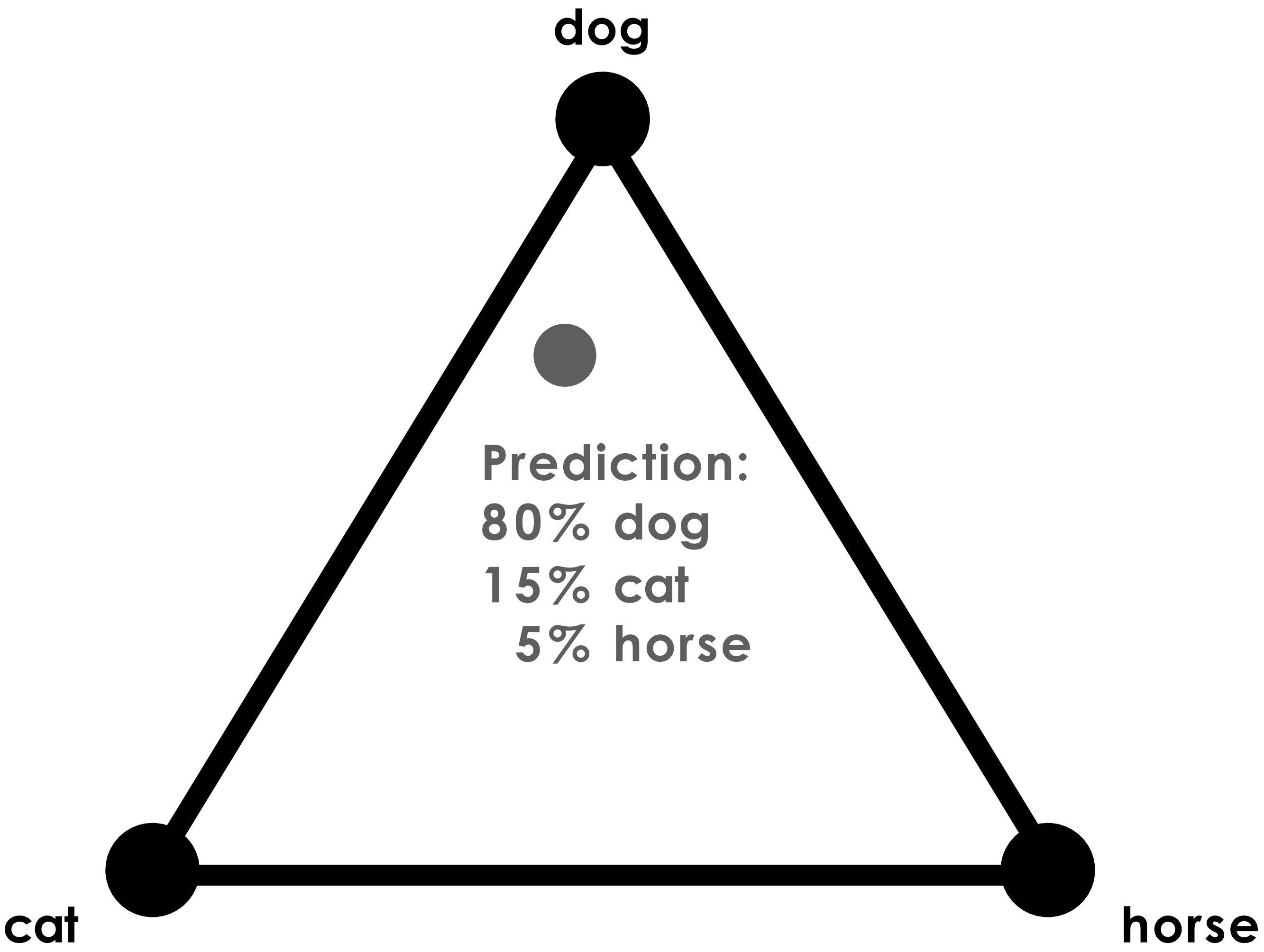
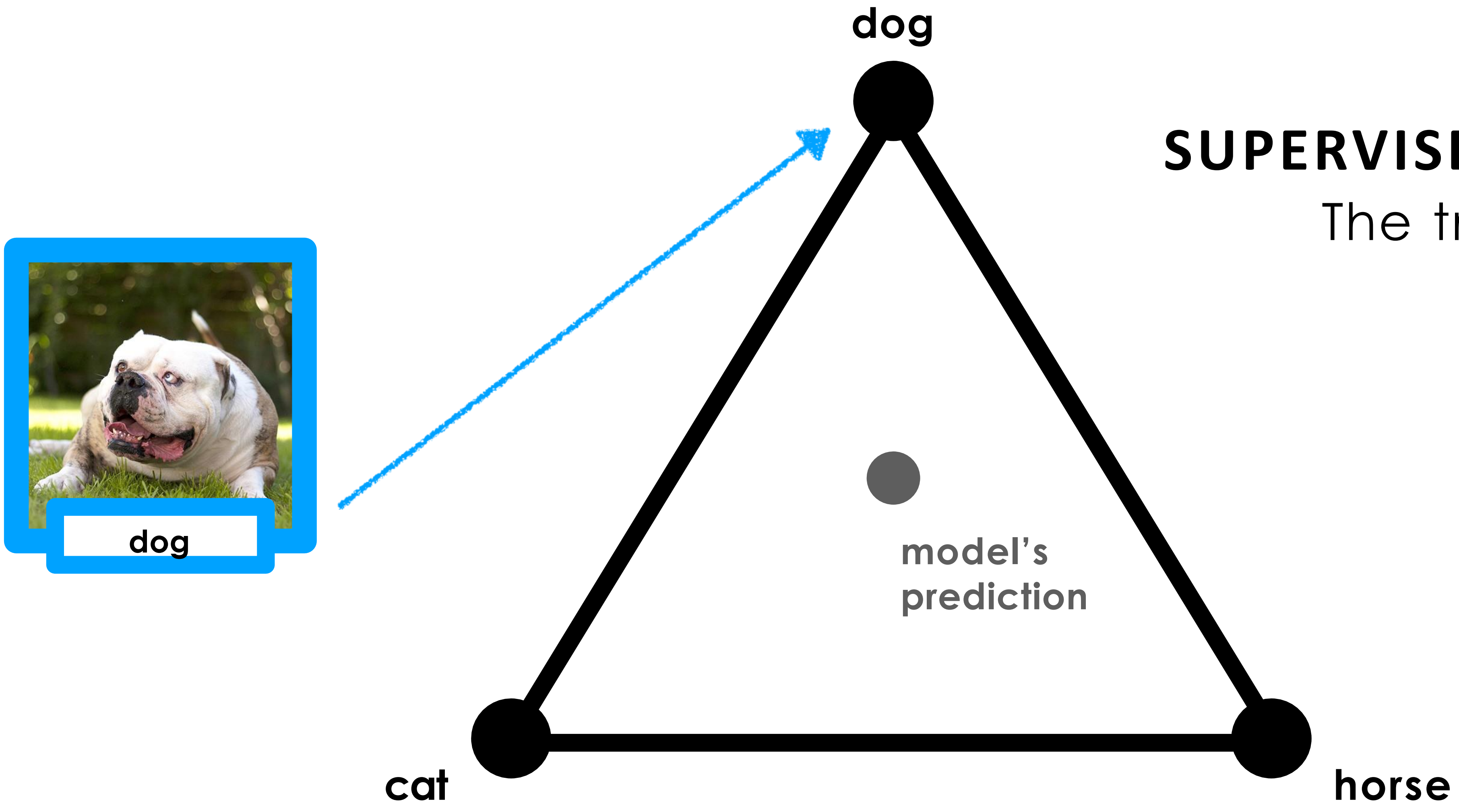


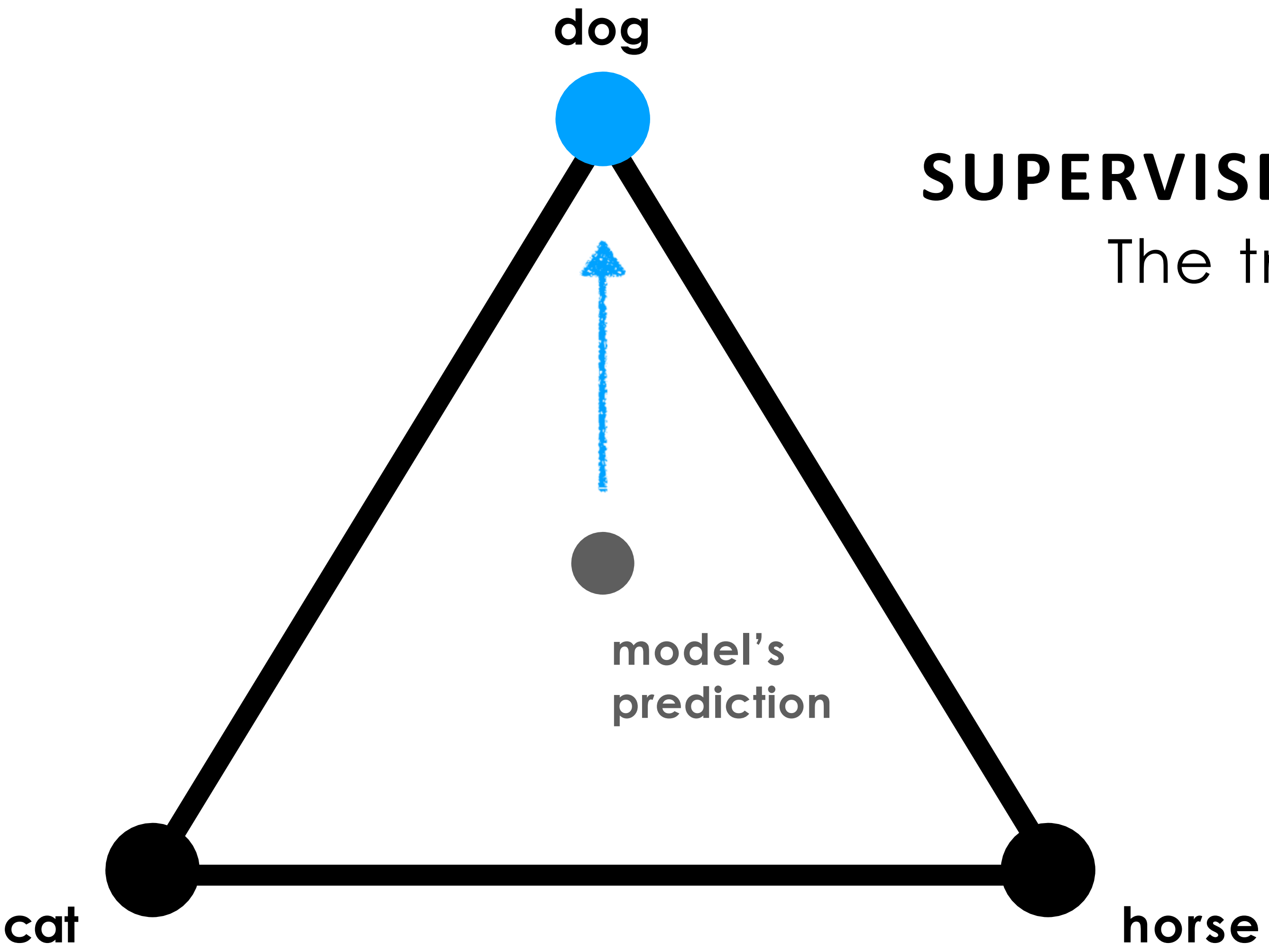
Illustration in the Probability Space



SUPERVISED LEARNING

The true label pulls predictions to its direction.

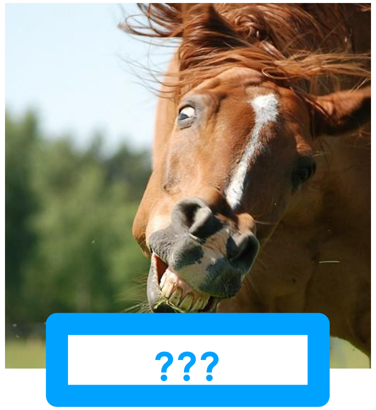
Illustration in the Probability Space

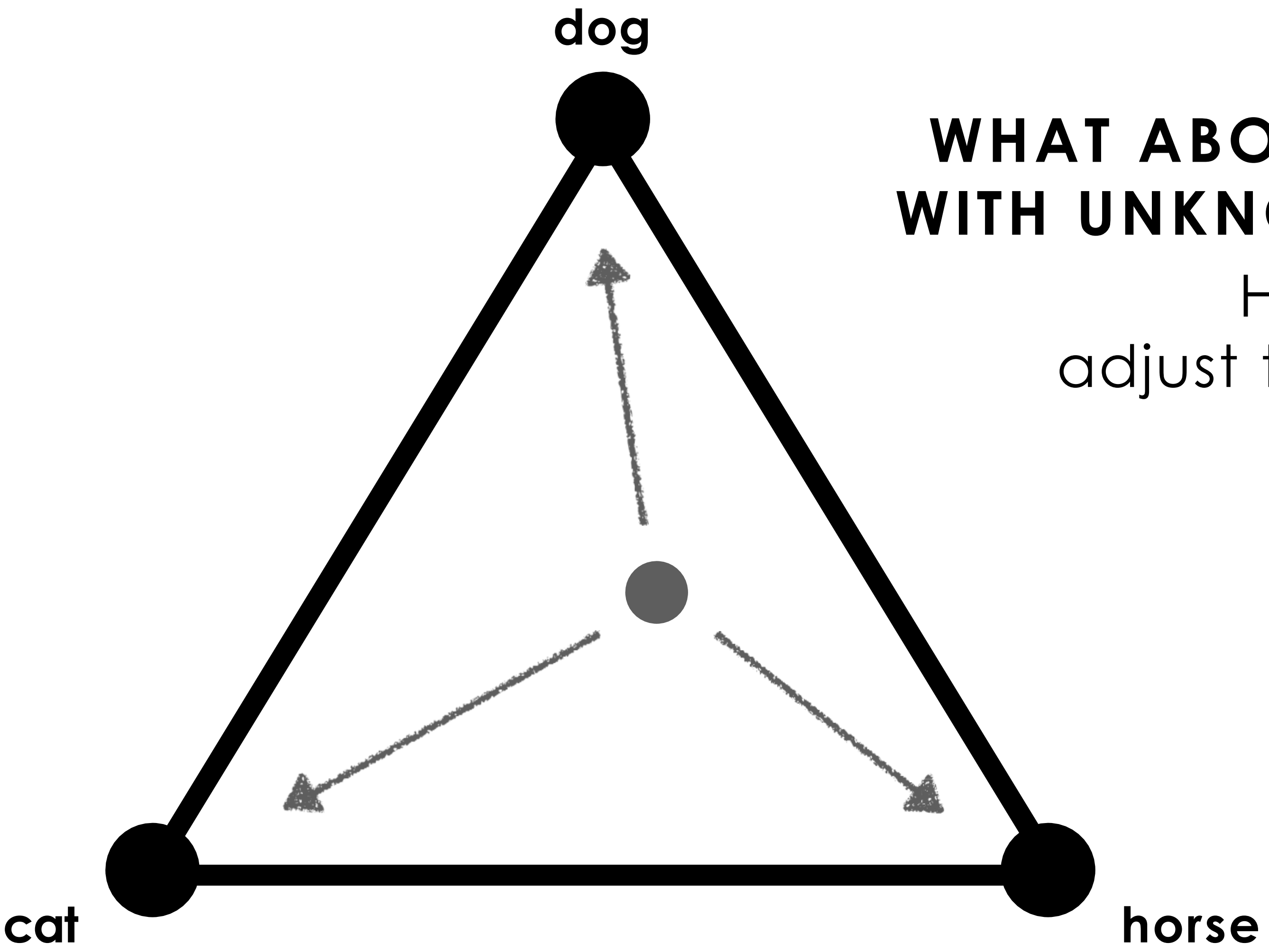


SUPERVISED LEARNING

The true label pulls predictions to its direction.

Learning from Unlabeled Images





WHAT ABOUT EXAMPLES WITH UNKNOWN LABELS?

How should we adjust the prediction?

SOTA Semi-supervised Learning Methods

Ladder Network (Γ Model) (Valpola et al., NIPS 2015)

Π model & Temporal Ensembling (Laine et al., ICLR 2017)

Mean Teacher (Tarvainen et al., NIPS 2017)

Virtual Adversarial Training (VAT) (Tarvainen et al., TPAMI 2018)

etc.

Unlabeled data can increase generalization

Generalization == to prevent **overfitting** on training data

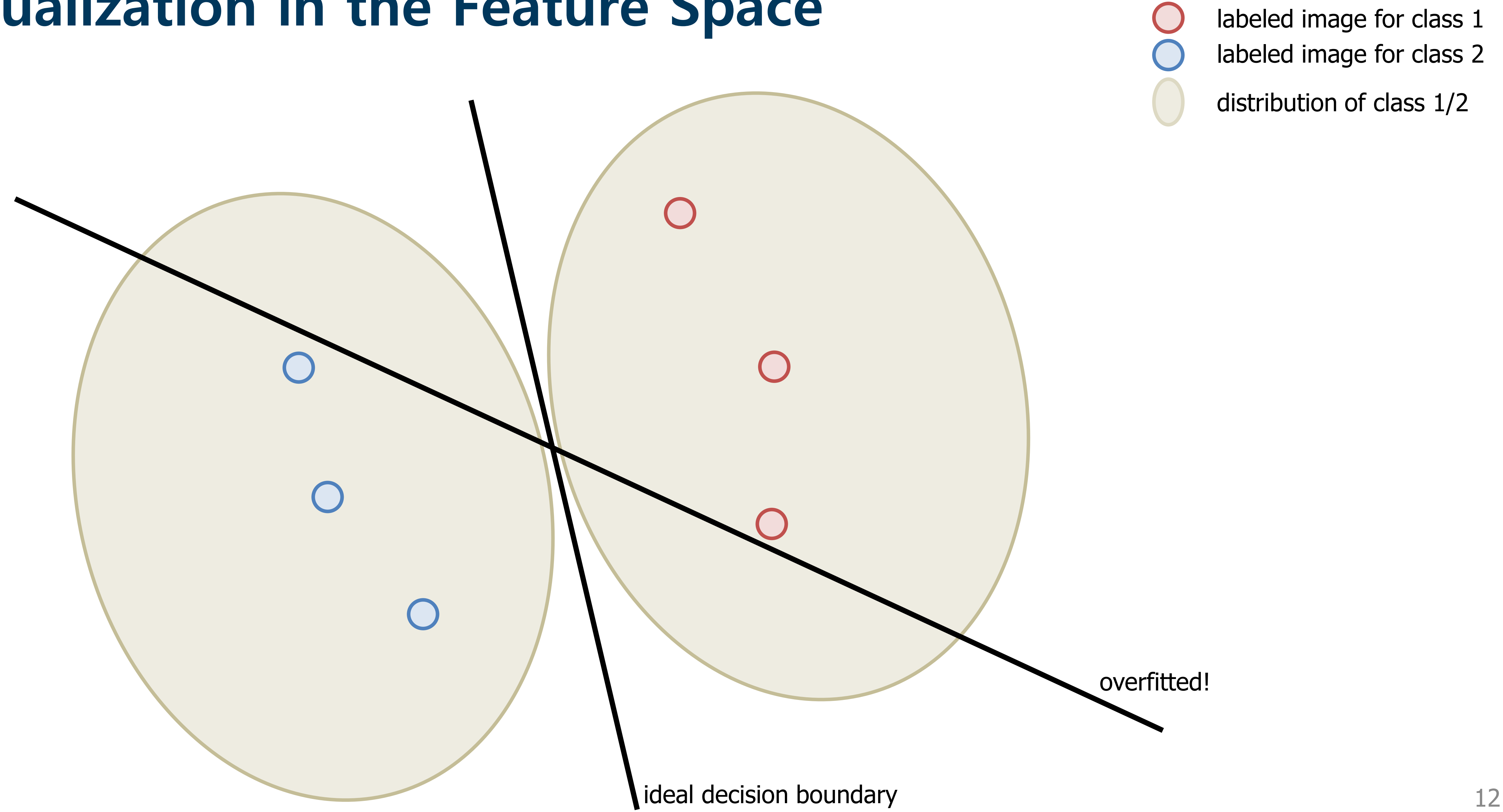
If the number (N_x) of dataset $\{x_i\}_{i=1}^{N_x}$ is **too few** to model the real data distribution $p(x)$,
The **overfitting** occurs.

By adding dense enough data, we can prevent overfitting.

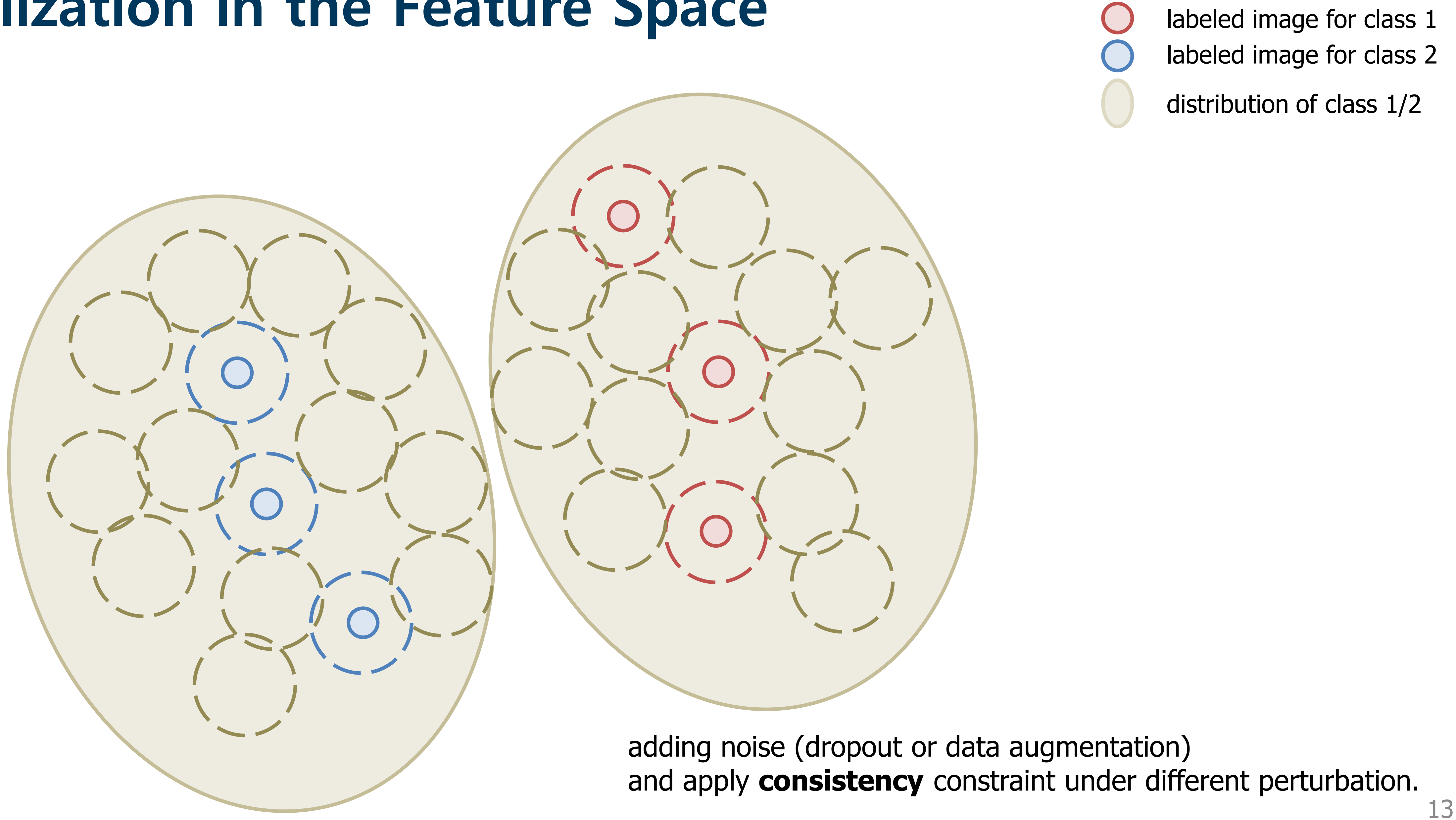
but labeling all the data might be cumbersome.

Semi-supervised learning is to
leverage dense data samples **without labels.**

Visualization in the Feature Space

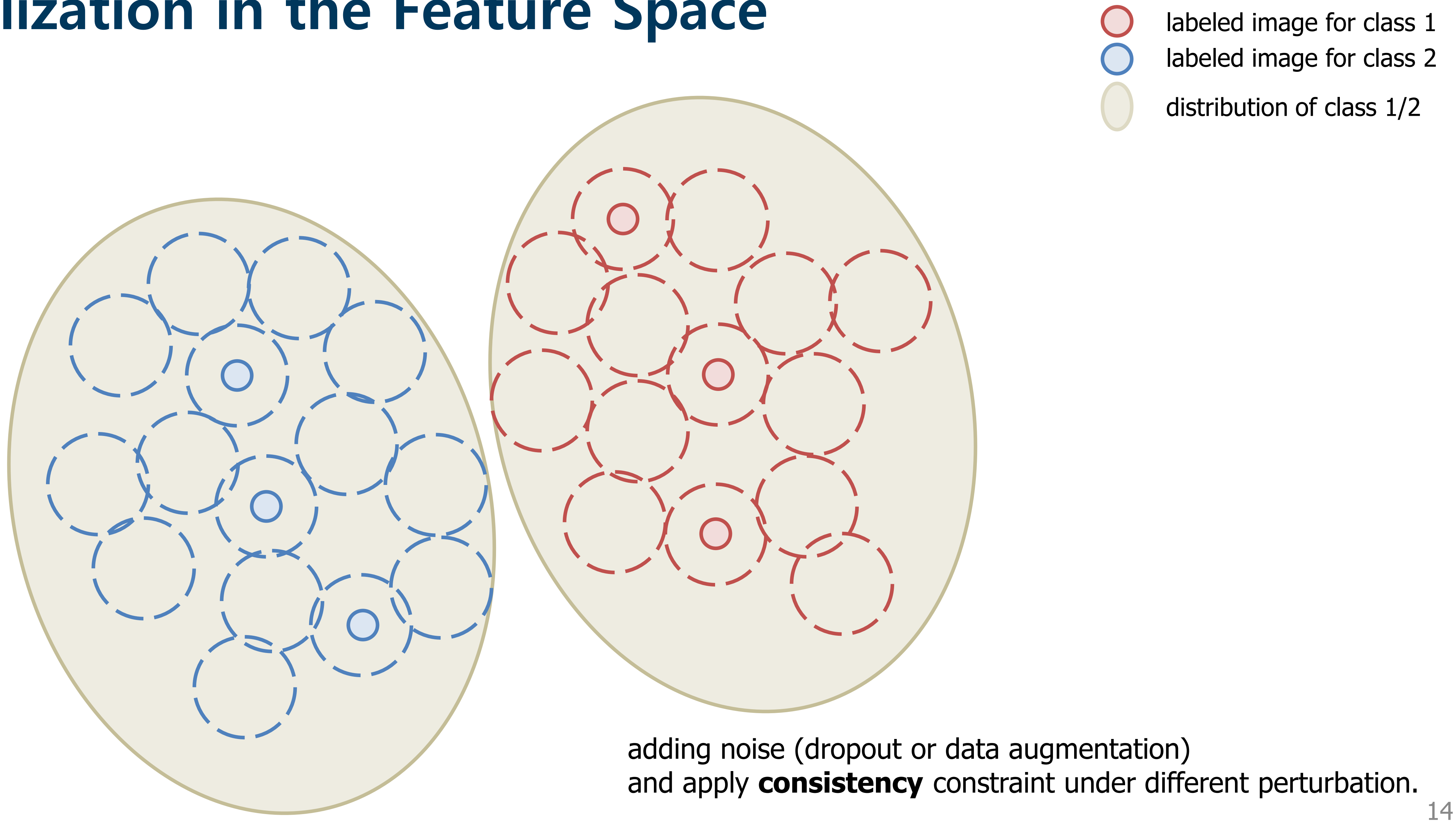


Visualization in the Feature Space



adding noise (dropout or data augmentation)
and apply **consistency** constraint under different perturbation.

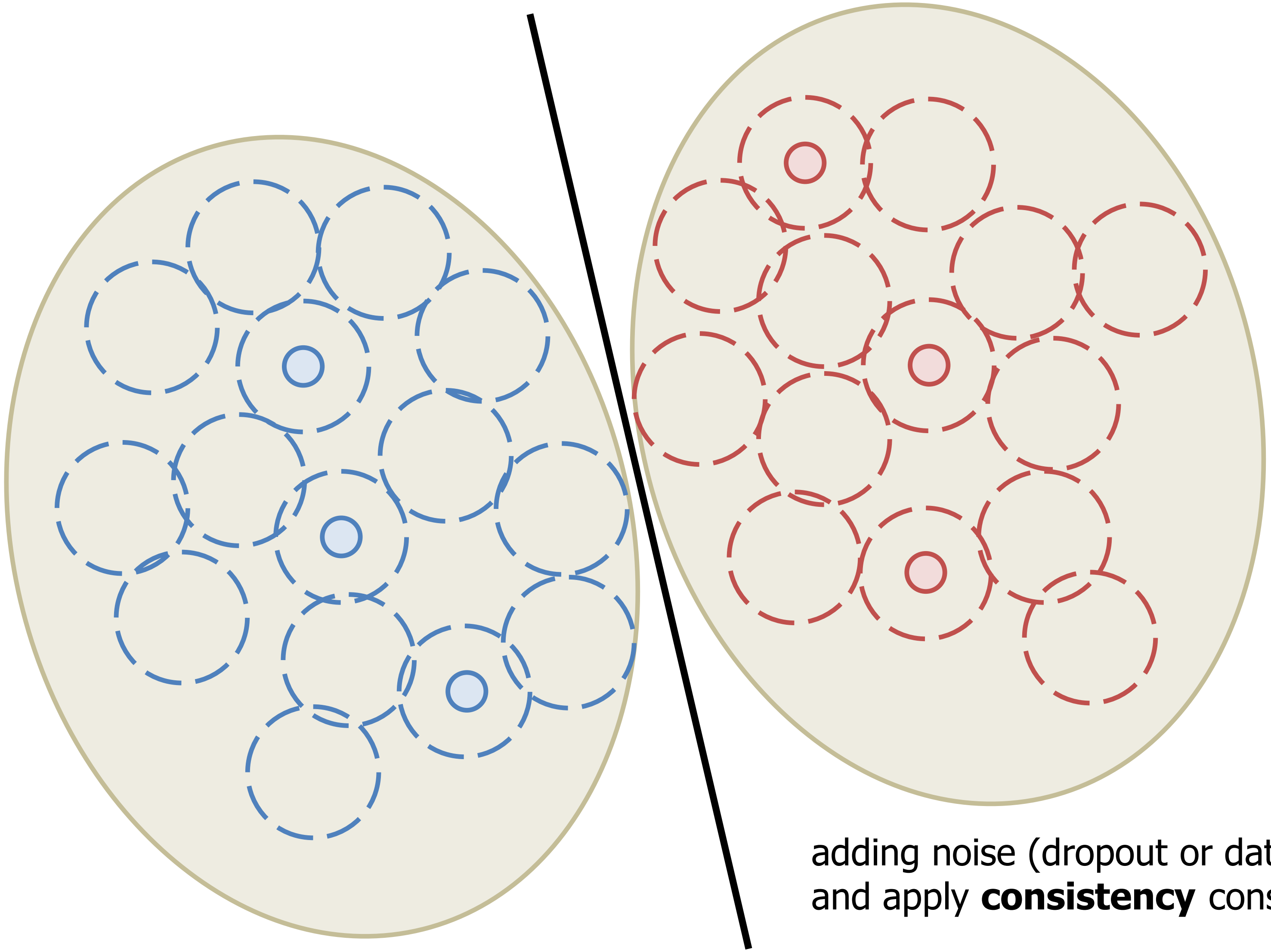
Visualization in the Feature Space



adding noise (dropout or data augmentation)
and apply **consistency** constraint under different perturbation.

Visualization in the Feature Space

labeled image for class 1
labeled image for class 2
distribution of class 1/2



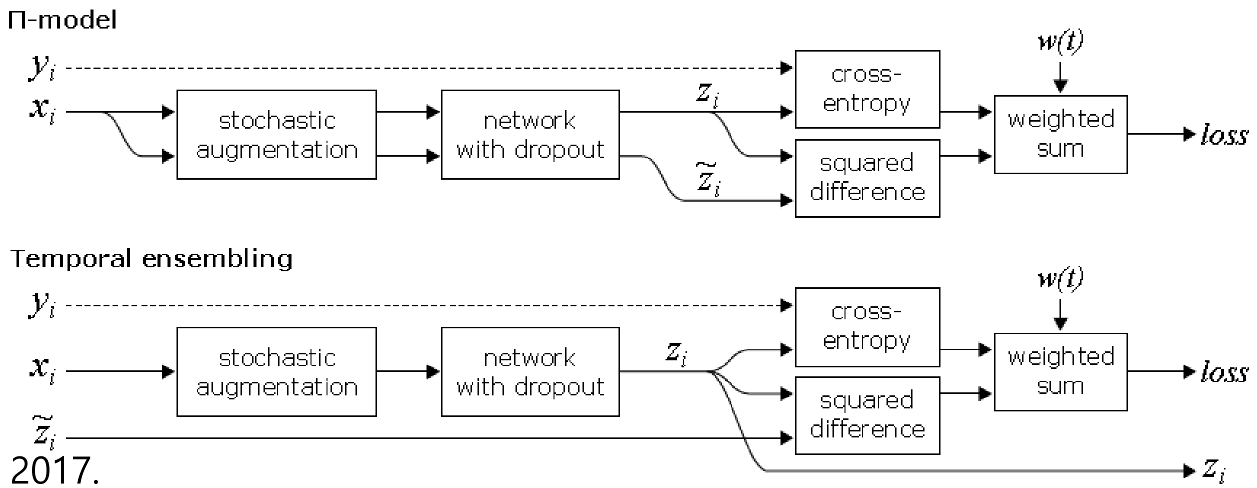
adding noise (dropout or data augmentation)
and apply **consistency** constraint under different perturbation.

Π model (Laine et al., ICLR 2017)

$$loss \leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap L)}\log z_i[y_i] + w(t)\frac{1}{C|B|}\sum_{i\in B}\underline{\|z_i - \tilde{z}_i\|}^2$$

Distance btw pred w/ different perturbations.

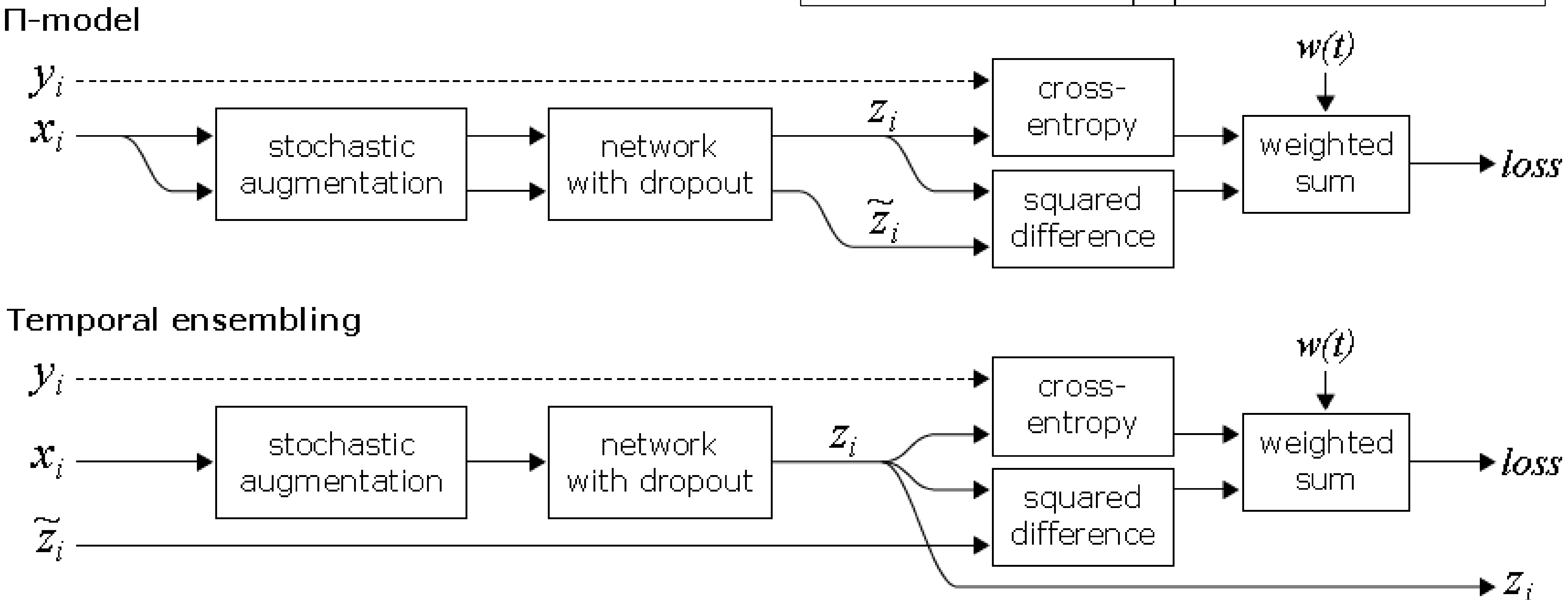
- Self-ensembling (exploiting drpout)
- Feed forward the **same model** under different (iid) perturbation (student, teacher)
- **Consistency** cost (L2 distance)
- To alleviate the **bias** of the teacher, add noise also to the teacher.
- Effect of the loss function : Minimizing variance of the prediction.
- Weakness : have to evaluate model twice.



[1] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.

Π model (Laine et al., ICLR 2017)

$$loss \leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap L)} \log z_i[y_i] + w(t)\frac{1}{C|B|}\sum_{i\in B}||z_i - \tilde{z}_i||^2$$



[1] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.

Temporal Ensembling

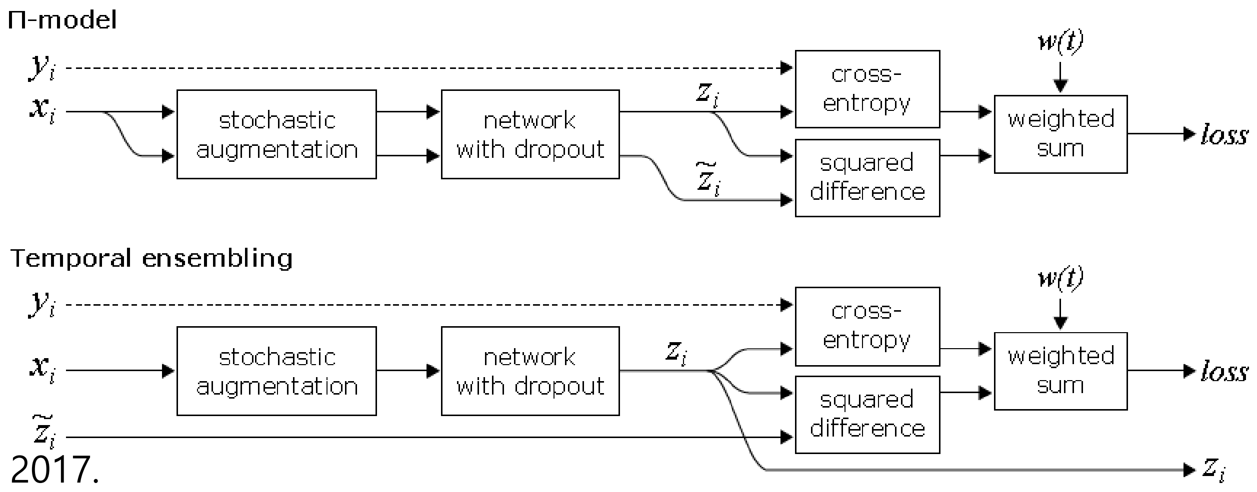
- Problem of Π model : teacher can be unstable.
- To reduce **variance** of targets, add **momentum** for teacher activation -> better (stable) teacher
- **Aggregate** all the previous activations with Exponential Moving Average (**EMA**)

$$\boxed{\tilde{F}^{(t)}(x_i) = \alpha \tilde{F}^{(t-1)}(x_i) + (1-\alpha) f^{(t)}(x_i; \theta, \xi)}$$

- As a target (**teacher**), we need debias correction.

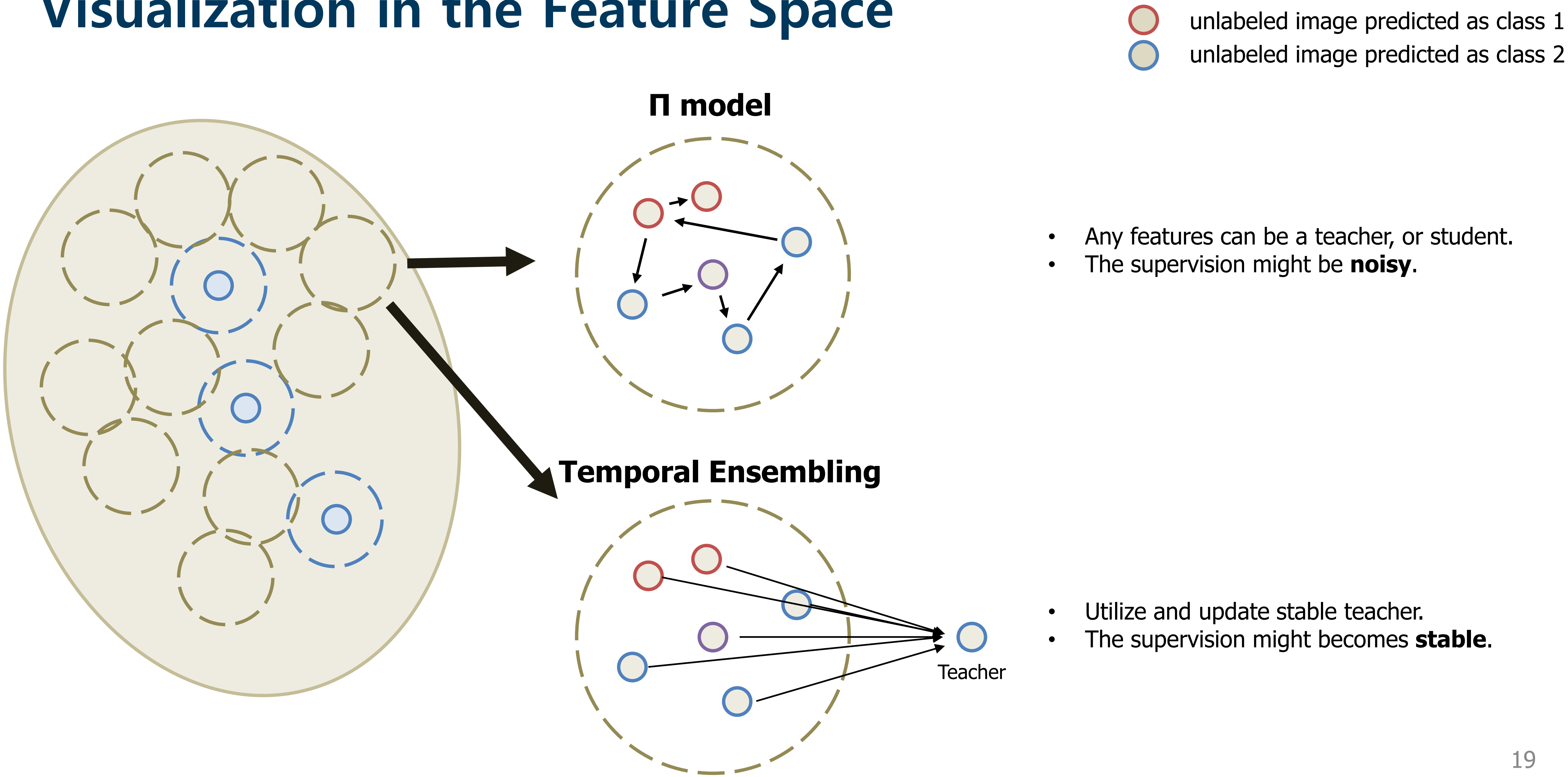
$$\boxed{\tilde{f}^{(t)}(x_i) = \tilde{F}^{(t)}(x_i)/(1-\alpha^t).}$$

- The EMA prediction (teacher) is an ensemble of the current and the earlier feats. (Temporal Ensemble)
- Weak : updating teacher only every **epoch**.



[1] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.

Visualization in the Feature Space



- Any features can be a teacher, or student.
- The supervision might be **noisy**.

- Utilize and update stable teacher.
- The supervision might becomes **stable**.

Mean Teacher (Tarvainen et al., NIPS 2017)

Improved version of temporal ensembling

Problem of Temporal Ensembling : teacher is updated every epoch -> **slow**, huge computation.

Instead of output activation, apply EMA for teacher **model's parameter**.

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

Updates teacher more frequently (every batch) -> better teacher.
The computation is independent to the number of samples.

More accurate target + enables learning large dataset

[1] MEAN TEACHERS ARE BETTER ROLE MODELS: Weight-averaged consistency targets improve semi-supervised deep learning results. NIPS 2017.

Further improvements with SNTG

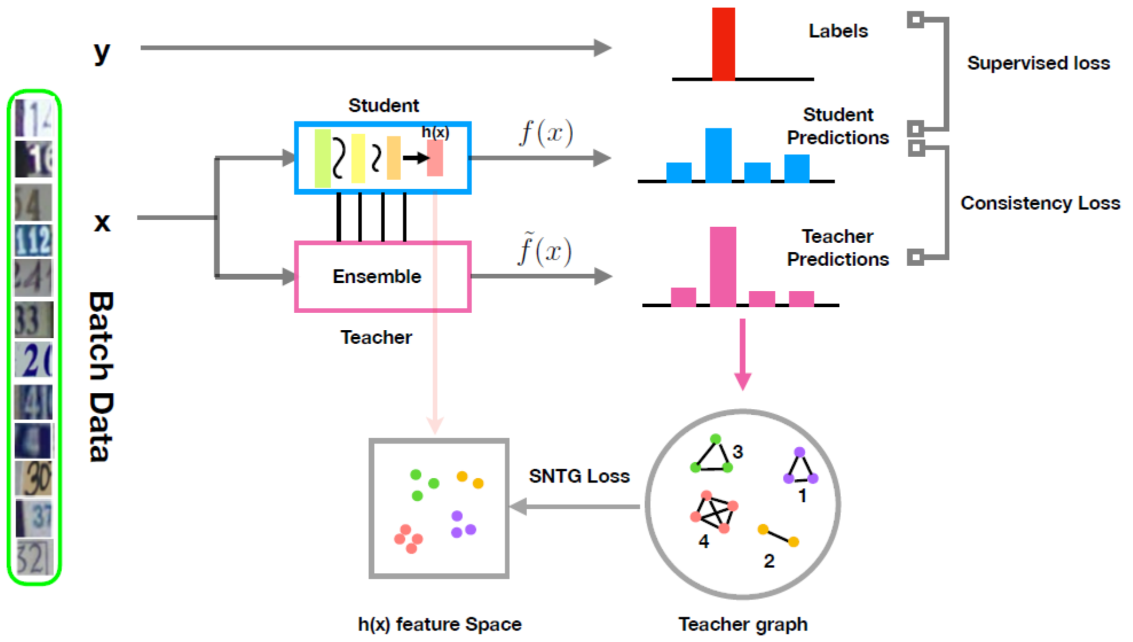
Smooth Neighbors on Teacher Graph (SNTG)
consider "**connections** between data points" to induce smoothness.
Construct a **similarity** graph (W) and use it as an additional supervision with SNTG loss.

Can be used for **boosting** the performance of Mean Teacher, Temporal Ensembling, etc.

$$W_{ij} = \begin{cases} 1 & \text{if } \tilde{y}_i = \tilde{y}_j \\ 0 & \text{if } \tilde{y}_i \neq \tilde{y}_j \end{cases}$$

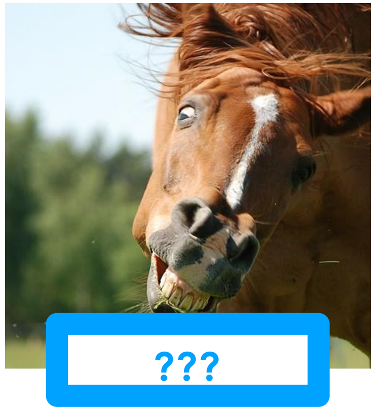
$$\boxed{R_S(\theta, \mathcal{L}, \mathcal{U}) = \sum_{x_i, x_j \in \mathcal{D}} \ell_G(h(x_i;\theta), h(x_j;\theta), W_{ij})}$$

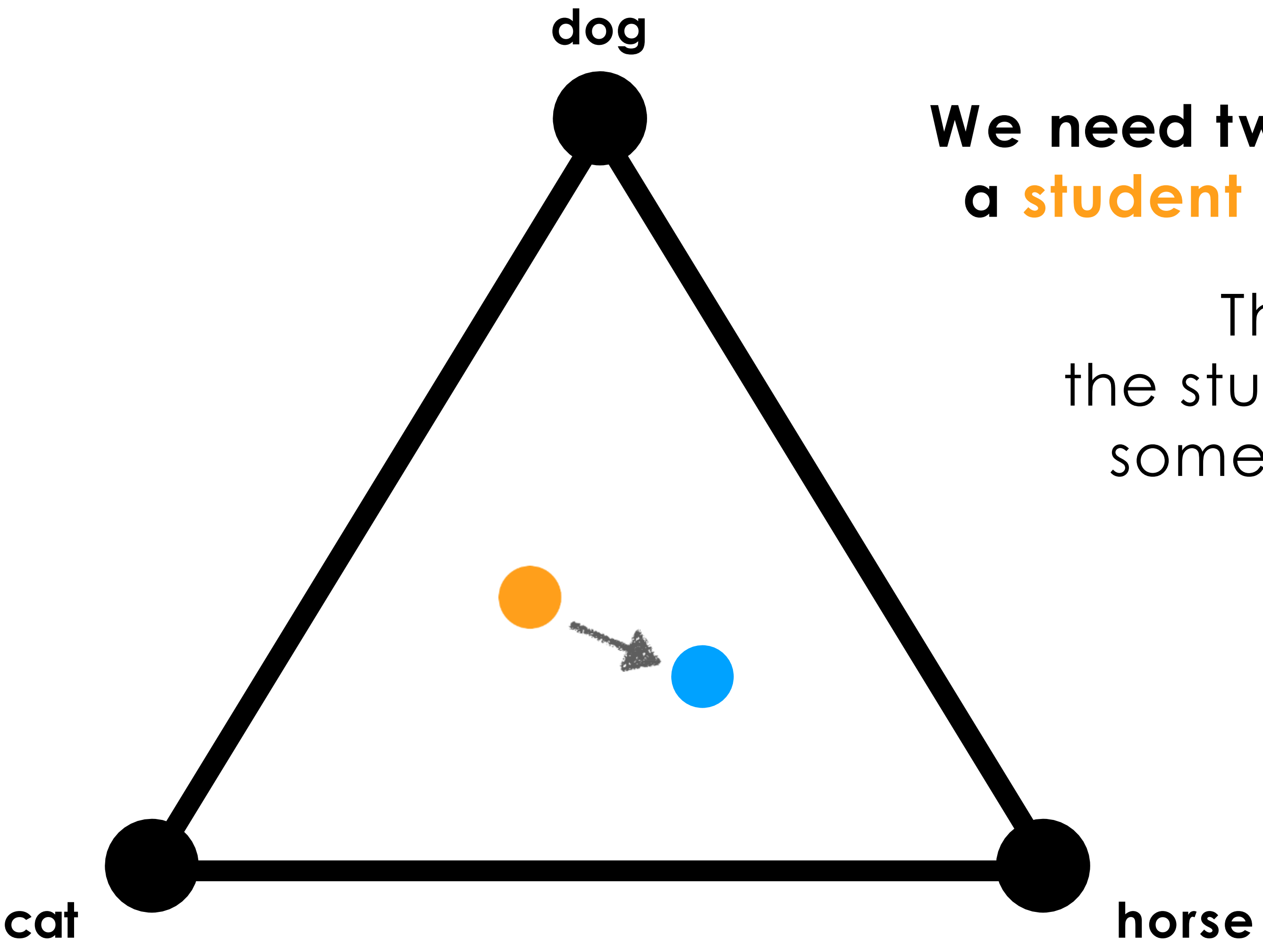
$$\ell_G = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max(0, m - \|h(x_i) - h(x_j)\|)^2 & \text{if } W_{ij} = 0 \end{cases}$$



[1] Smooth Neighbors on Teacher Graphs for Semi-supervised Learning. CVPR 2018.

Learning from Unlabeled Images

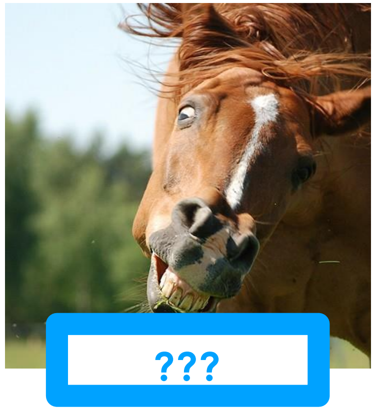


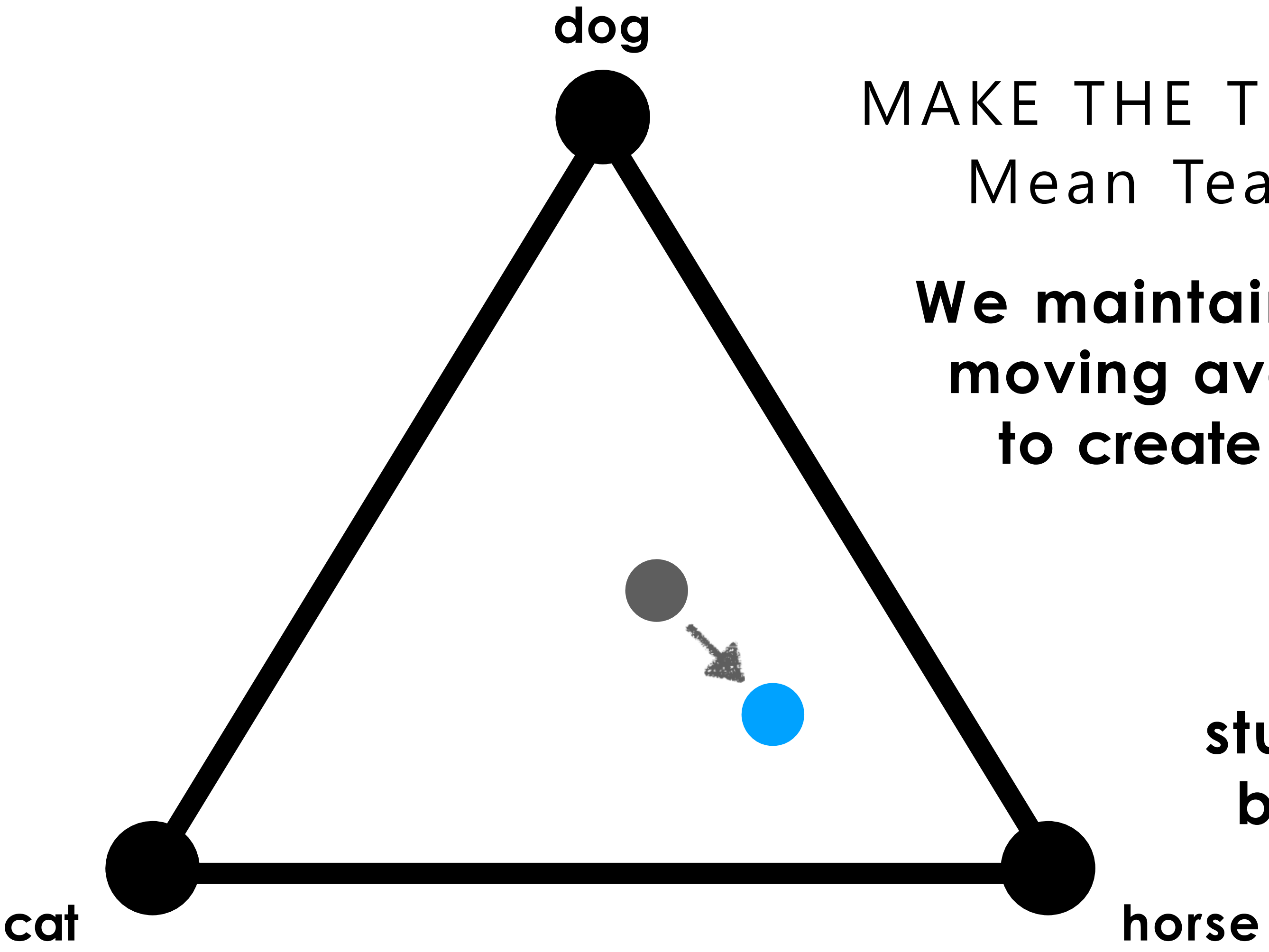


We need two predictions: a student and a teacher.

Then, hopefully, the student can learn something from the teacher.

Learning from Unlabeled Images





MAKE THE TEACHER BETTER.

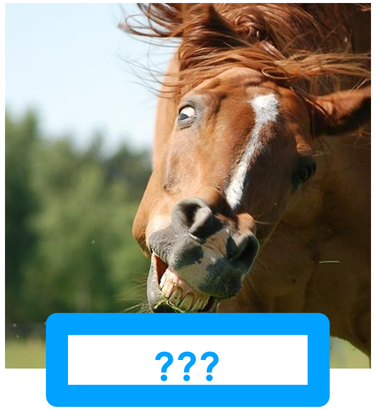
Mean Teacher (Proposed)

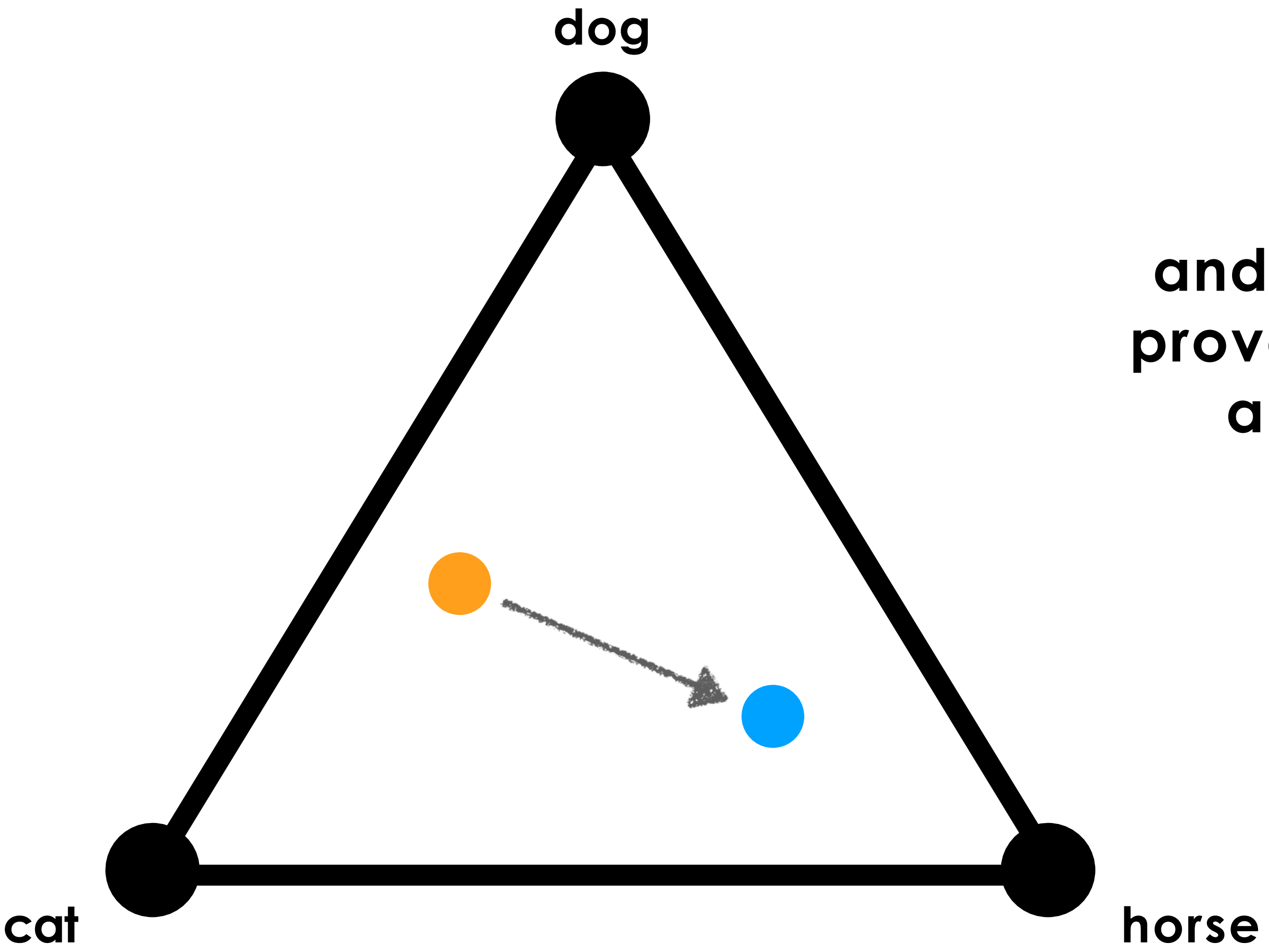
We maintain an exponential moving average of weights to create a better teacher.

A *mean* teacher.

Then we let the student learn these better predictions.

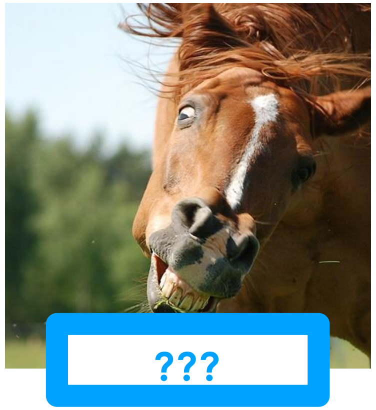
Learning from Unlabeled Images

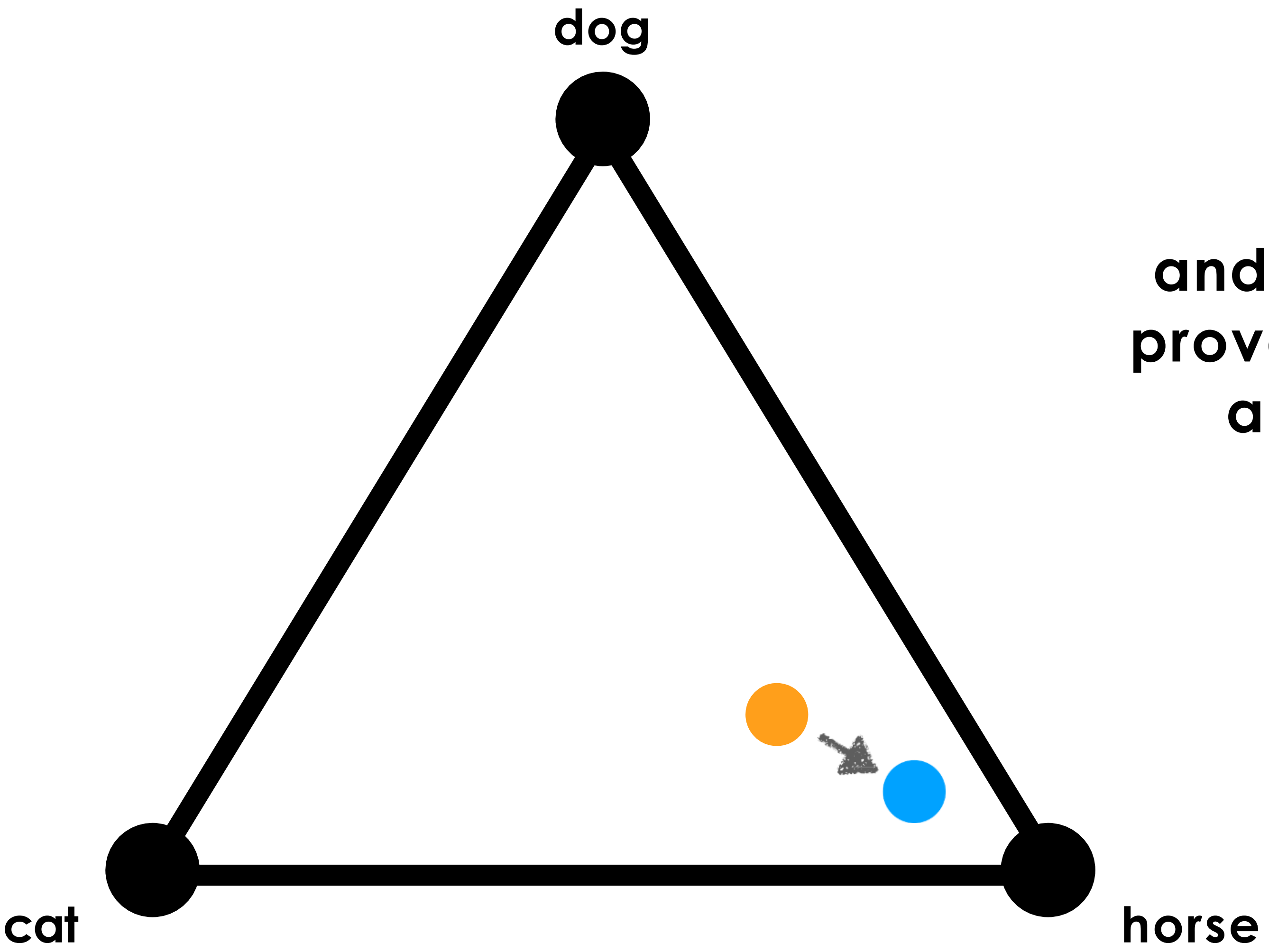




The student
and the teacher im
prove each other in
a virtuous cycle.

Learning from Unlabeled Images

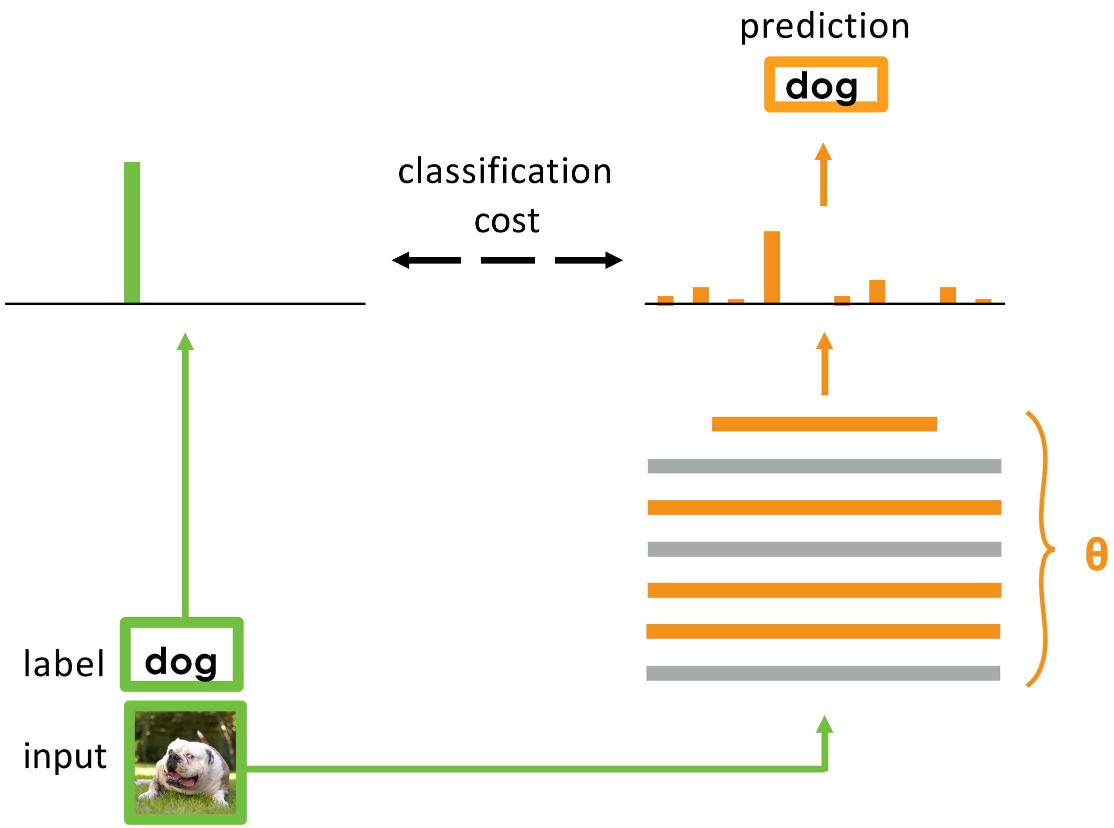




The student and the teacher improve each other in a virtuous cycle.

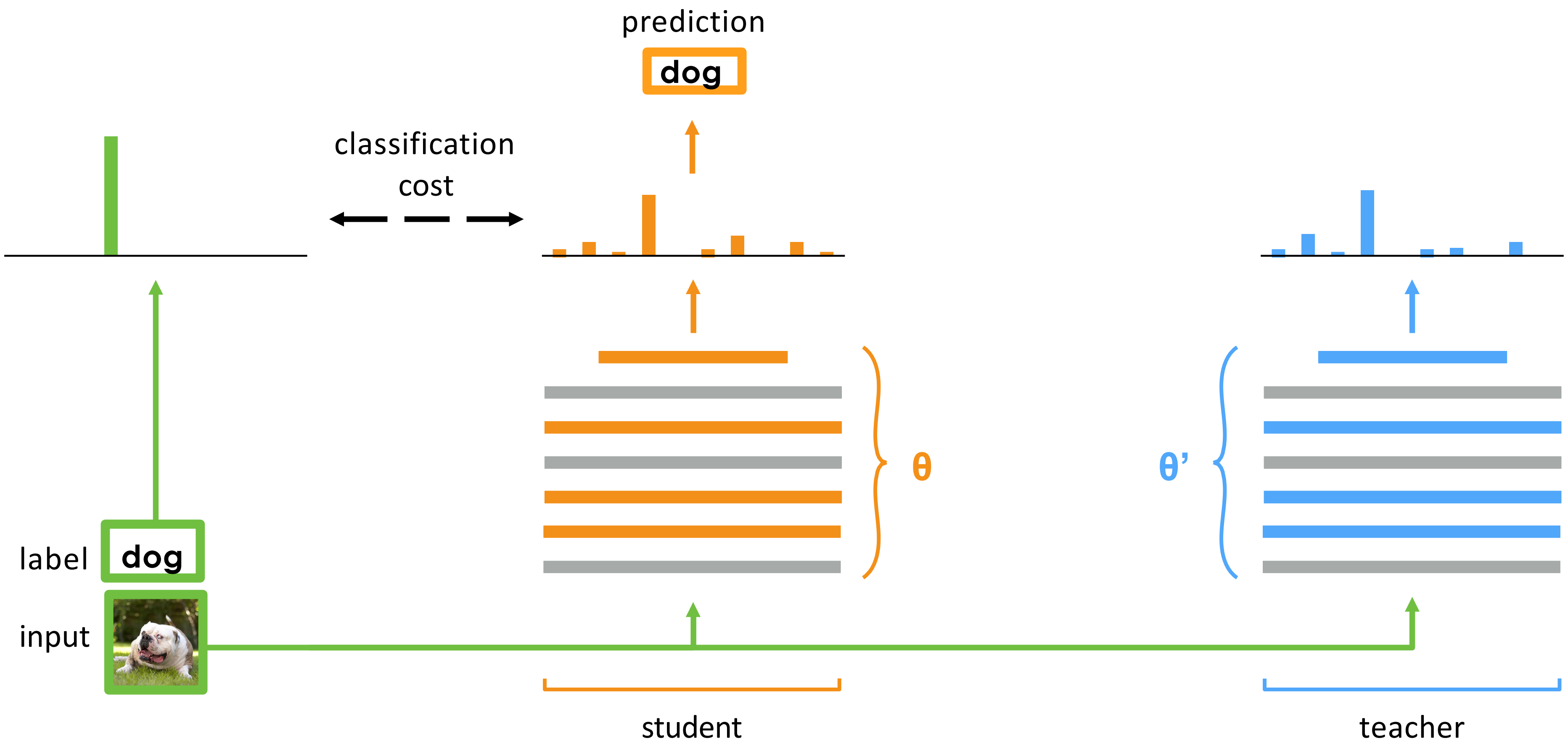
Mean Teacher

Take a supervised model.



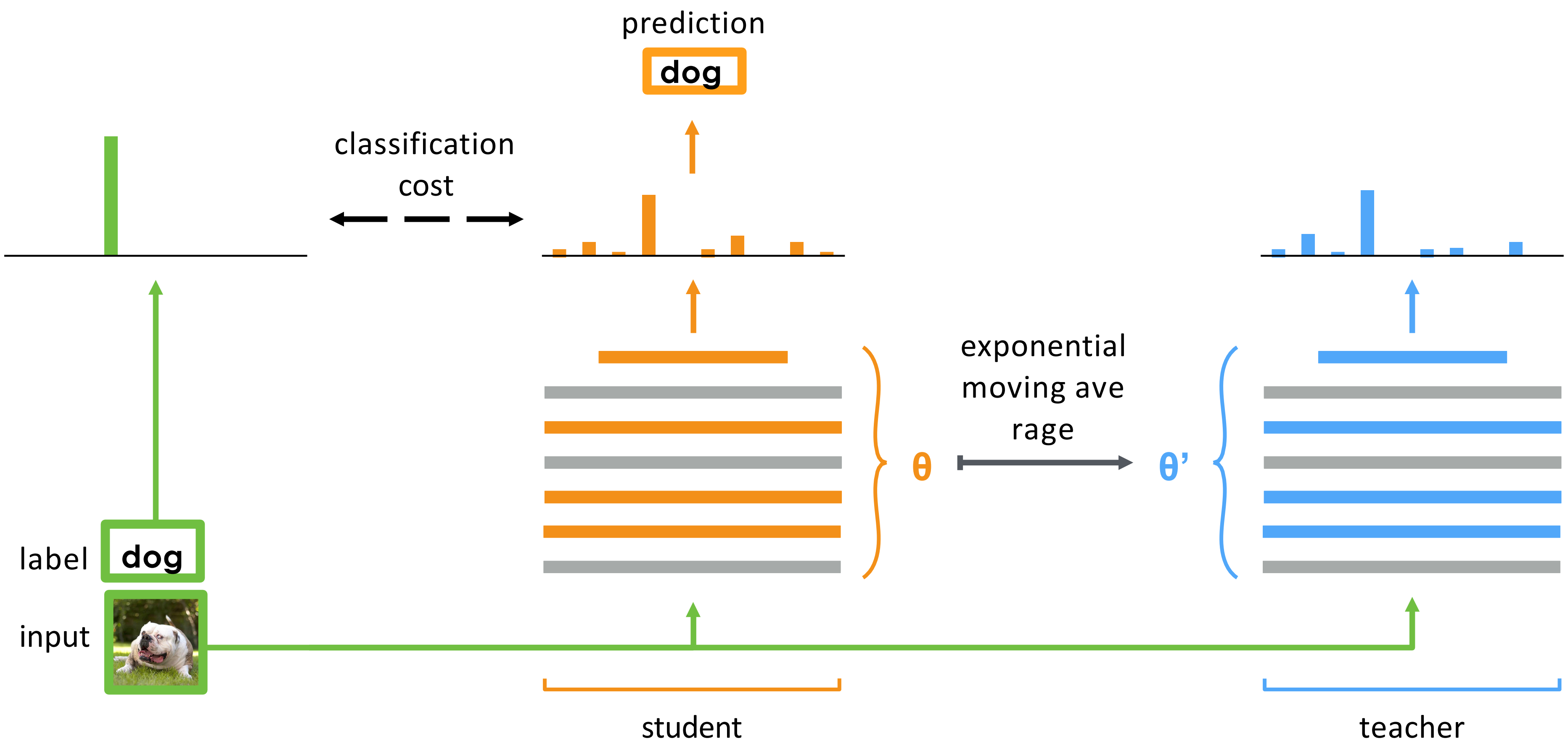
Mean Teacher

Make a copy of it.



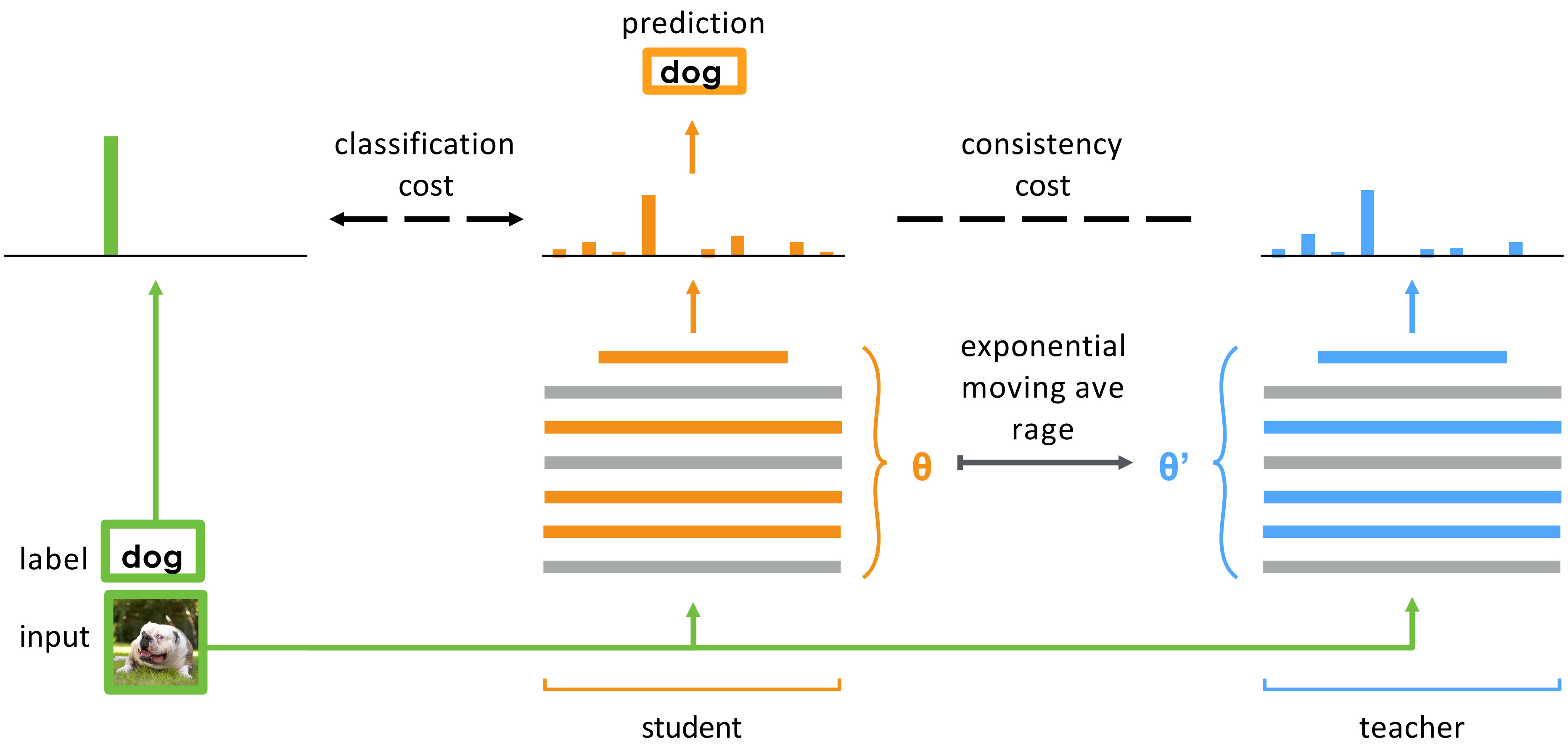
Mean Teacher

Update teacher weights after each training step.



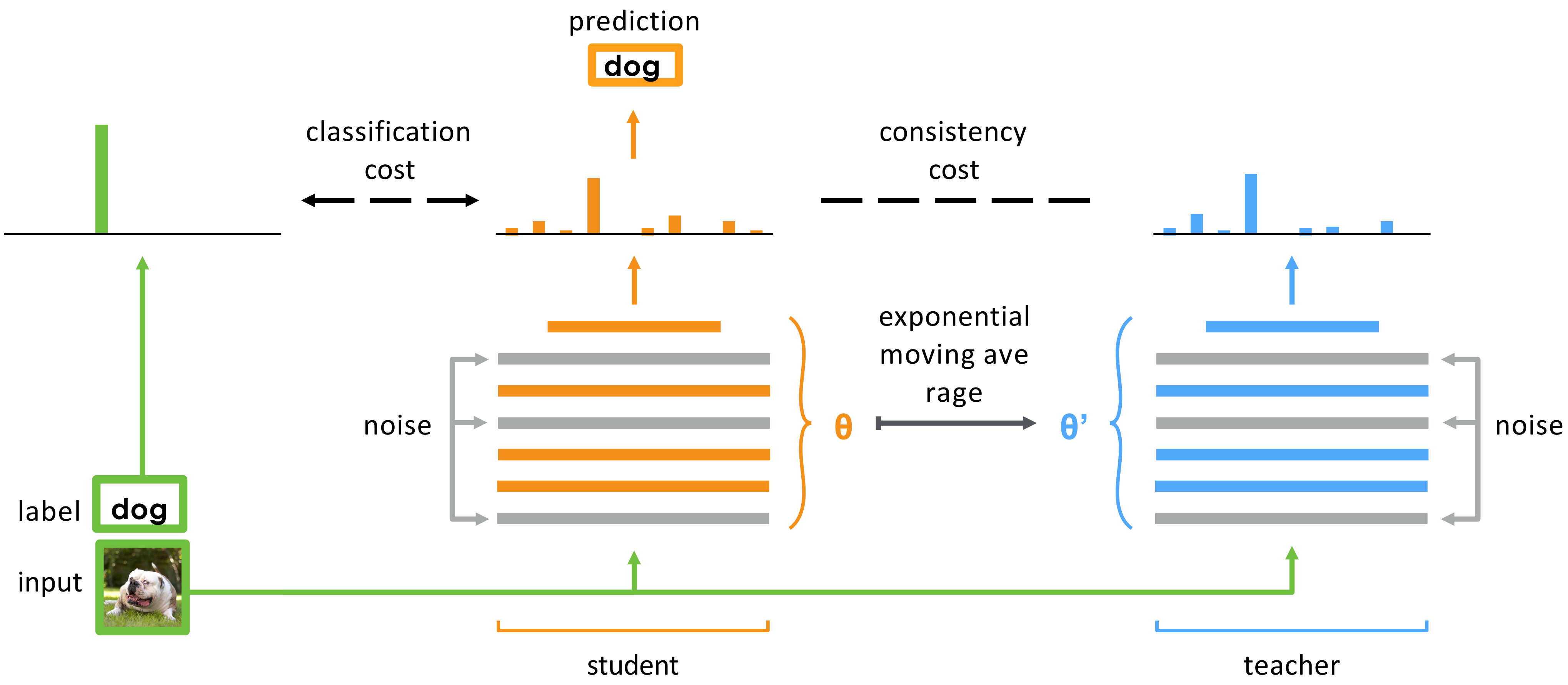
Mean Teacher

Add a cost between the two predictions.



Mean Teacher

Maybe add some noise.



Comparison to other methods on SVHN and CIFAR-10

	250 labels 73257 images	500 labels 73257 images	1000 labels 73257 images	73257 labels 73257 images
GAN [24]		18.44 ± 4.8	8.11 ± 1.3	
Π model [13]		6.65 ± 0.53	4.82 ± 0.17	2.54 ± 0.04
Temporal Ensembling [13]		5.12 ± 0.13	4.42 ± 0.16	2.74 ± 0.06
VAT+EntMin [15]			**3.86**	
Supervised-only	27.77 ± 3.18	16.88 ± 1.30	12.32 ± 0.95	2.75 ± 0.10
Π model	9.69 ± 0.92	6.83 ± 0.66	4.95 ± 0.26	2.50 ± 0.07
Mean Teacher	$\mathbf{4.35 \pm 0.50}$	$\mathbf{4.18 \pm 0.27}$	3.95 ± 0.19	$\mathbf{2.50 \pm 0.05}$

SVHN

	1000 labels 50000 images	2000 labels 50000 images	4000 labels 50000 images	50000 labels 50000 images
GAN [24]			18.63 ± 2.32	
Π model [13]			12.36 ± 0.31	5.56 ± 0.10
Temporal Ensembling [13]			12.16 ± 0.31	$\mathbf{5.60 \pm 0.10}$
VAT+EntMin [15]			**10.55**	
Supervised-only	46.43 ± 1.21	33.94 ± 0.73	20.66 ± 0.57	5.82 ± 0.15
Π model	27.36 ± 1.20	18.02 ± 0.60	13.20 ± 0.27	6.06 ± 0.11
Mean Teacher	$\mathbf{21.55 \pm 1.48}$	$\mathbf{15.73 \pm 0.31}$	12.31 ± 0.28	5.94 ± 0.15

CIFAR10

Instruction for PA#3

0. Environment setting

Go to the link (https://pytorch.org/) and install **Pytorch**.

Regarding your machine configuration (e.g. CUDA version), choose the appropriate command to install the latest Pytorch version.

QUICK START
LOCALLY

Select your preferences and run the install command. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

PyTorch Build	**Stable**	Preview			
Your OS	**Linux**	Mac	Windows		
Package	Conda	**Pip**	LibTorch	Source	
Language	**Python 2.7**	Python 3.5	Python 3.6	Python 3.7	C++
CUDA	8.0	**9.0**	9.2	None	
Run this Command:	`pip install torch torchvision`				

1. Data Setup

Target dataset : **CIFAR10** data

Randomly select **4000** samples among the training set as labeled,
and make the rest as unlabeled.

Refer Pytorch image classification tutorial for settings.
(https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html).

For semi-supervised learning setup, please refer to the recent works [1,2]

[1] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.
[2] MEAN TEACHERS ARE BETTER ROLE MODELS: Weight-averaged consistency targets improve semi-supervised deep learning results. NIPS 2017.

2. Network Implementation.

Build ConvLarge architecture [1]
which is the benchmark architecture proposed for semi-supervised learning.

Construct your base network given **this description.** →

NAME	DESCRIPTION
input	32×32 RGB image
noise	Additive Gaussian noise $\sigma = 0.15$
conv1a	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv1b	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv1c	128 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
pool1	Maxpool 2×2 pixels
drop1	Dropout, $p = 0.5$
conv2a	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv2b	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
conv2c	256 filters, 3×3, pad = 'same', LReLU ($\alpha = 0.1$)
pool2	Maxpool 2×2 pixels
drop2	Dropout, $p = 0.5$
conv3a	512 filters, 3×3, pad = 'valid', LReLU ($\alpha = 0.1$)
conv3b	256 filters, 1×1, LReLU ($\alpha = 0.1$)
conv3c	128 filters, 1×1, LReLU ($\alpha = 0.1$)
pool3	Global average pool ($6 \times 6 \rightarrow 1 \times 1$ pixels)
dense	Fully connected $128 \rightarrow 10$
output	Softmax

Required Library : torch.nn & torch.nn.functional

Refer [1] for more detailed design choices.

"The ones who use different architecture will be penalized."

[1] TEMPORAL ENSEMBLING FOR SEMI-SUPERVISED LEARNING.ICLR 2017.

3. Training a Supervised Model.

- Define **cross-entropy** loss as supervised loss.
- compute the supervised loss and update the network with **Adam** optimizer.
- **Report** performance of the model trained "only on supervised data". (1)
- This will be your baseline performance.
- Required library: torchvision, torch.optim

4. Implement Mean teacher

1. Make copy of the student model as a **teacher** model.
2. Given a training batch, compute the **consistency** loss between teacher and student model.

$$J(\theta) = \mathbb{E}_{x,\eta',\eta}\left[\|f(x,\theta',\eta') - f(x,\theta,\eta)\|^2\right]$$

3. Train the **student** with supervised loss and consistency loss via gradient descent.
4. Update the **teacher** with exponential moving average (EMA).

$$\theta'_t = \alpha\theta'_{t-1} + (1-\alpha)\theta_t$$

5. Repeat 2-4.

Report performance of the "Mean Teacher (MT) model". (2)

Refer the paper [1] for more details.

[1] MEAN TEACHERS ARE BETTER ROLE MODELS: Weight-averaged consistency targets improve semi-supervised deep learning results. NIPS 2017.

5. Implement SNTG

1. Construct Teacher Graph between the data points.

$$W_{ij} = \begin{cases} 1 & \text{if } \tilde{y}_i = \tilde{y}_j \\ 0 & \text{if } \tilde{y}_i \neq \tilde{y}_j \end{cases}$$

2. Compute SNTG loss between the latent features.

$$\ell_G = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max(0, m - \|h(x_i) - h(x_j)\|)^2 & \text{if } W_{ij} = 0 \end{cases}$$

3. Update SNTG loss along with mean teacher loss.

Report performance of "MT+SNTG model". (3)

Refer the paper [1] for more details.

Algorithm 1 Mini-batch training of SNTG for SSL

Require: x_i = training inputs, y_i for labeled inputs in $\mathcal{L}$
Require: $w(t)$ = unsupervised weight ramp-up function
Require: $f_\theta(x)$ = neural network with parameters θ

```
for t in [1, numepochs] do
    for each minibatch B do
        f_i ← f_θ(x_{i∈B}) evaluate network outputs
        f̃_i ← f̃(x_{i∈B}) given by the teacher model
        for (x_i, x_j) in a minibatch pairs S from B do
            Compute W_ij according to Eq. (6)
        end for
        loss ← −(1/|B|) Σ_{i∈(B∩L)} log[f_i]_{y_i}
            +w(t)[λ_1 (1/|B|) Σ_{i∈B} d(f̃_i, f_i)
                +λ_2 (1/|S|) Σ_{i,j∈S} ℓ_G(h(x_i), h(x_j), W_ij)]
        update θ using optimizers, e.g., Adam [23]
    end for
end for
return θ
```

[1] Smooth Neighbors on Teacher Graphs for Semi-supervised Learning, CVPR 2018.

6. Extra Credit (Implement the Latest Approaches)

Implement one of the recent works.

- Deep Co-Training for Semi-Supervised Image Recognition (ECCV 18)
- Transductive Semi-Supervised Deep Learning using Min-Max Features (ECCV 18)
- Semi-Supervised Deep Learning with Memory (ECCV 18)
- SaaS: Speed as a Supervisor for Semi-supervised Learning (ECCV 18)
- HybridNet: Classification and Reconstruction Cooperation for Semi-Supervised Learning (ECCV 18)
- ……

Report performance of "your final model ". (4)

For final performance, try any tricks to improve the performance.
(e.g. hyper-parameter tuning, different data augmentation, different noise)

but don't change the architecture (e.g. to ResNet)

"If you apply your own idea, you will receive higher score"

Criteria

- Implement a baseline model [1.0]
- Implement and reproduce Mean Teacher method [0.5]
- Implement Smooth Neighbor Teacher Graph [0.5]
- Improving Performance [1.0 (+1.0 for novelty)]

- Total 3.0+1.0 points.

"Performance is the most important criterion!
Try your best to improve the scores."

Submission

Due : December 7th (23:59 PM)
Use Pytorch library

Running code + report (description + performance)

Submit zip file to TA (djnjusa@kaist.ac.kr).
Zip file format : PA3_studentID_yourNAME.zip

***"The submission after deadline will not be accepted.
Just submit what you have done."***